# Package 'EGAD'

July 12, 2023

**Type** Package

**Title** Extending guilt by association by degree

**Version** 1.29.0

**Date** 2016-04-20

**Description** The package implements a series of highly efficient tools to calculate functional properties of networks based on guilt by association methods.

**License** GPL-2

**Depends** R(>= 3.5)

**Imports** gplots, Biobase, GEOquery, limma, impute, RColorBrewer, zoo, igraph, plyr, MASS, RCurl, methods

**Suggests** knitr, testthat, rmarkdown, markdown

**VignetteBuilder** rmarkdown

**RoxygenNote** 7.1.1

**LazyData** true

**LazyDataCompression** gzip

**Encoding** UTF-8

**biocViews** Software, FunctionalGenomics, SystemsBiology, GenePrediction, FunctionalPrediction, NetworkEnrichment, GraphAndNetwork, Network

**Author** Sara Ballouz [aut, cre], Melanie Weber [aut, ctb], Paul Pavlidis [aut], Jesse Gillis [aut, ctb]

**Maintainer** Sara Ballouz <sarahballouz@gmail.com>

**git_url** https://git.bioconductor.org/packages/EGAD

**git_branch** devel

**git_last_commit** 6ba1a84

**git_last_commit_date** 2023-04-25

**Date/Publication** 2023-07-12

# R topics documented:

---

`assortativity` *Calculating network assortativity*

---

## Description

The function calculates the assortativity of a network, that measures the preference of interactions between similar nodes. As in most literature, 'similarity' is here defined in terms of node degrees.

## Usage

```
assortativity(network)
```

## Arguments

| | |
|---|---|
| `network` | matrix indicating network structure (symmetric) |

## Value

Numeric value

## Examples

```
network <- matrix( sample(c(0,1),36, replace=TRUE), nrow=6,byrow=TRUE)
assort_value <- assortativity(network)
```

## attr.human *Human GENCODE annotations (v22)*

### Description

A dataset containing identifiers for gene transcripts

### Format

A data frame with 60483 rows and 10 variables:

**chr** chromosome

**start** chromosomal start position, in base pairs

**end** chromosomal end position, in base pairs

**strand** chromosomal strand, + or -

**un** unknown

**ensemblID** ENSEMBL identifier

**type** type of transcript

**stat** status of transcript

**name** HUGO identifier

**entrezID** Entrez identifier

@source ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_human/release_22/

## attr.mouse *Mouse GENCODE annotations (M7)*

### Description

A dataset containing identifiers for gene transcripts

### Format

A data frame with 46517 rows and 10 variables:

**chr** chromosome

**start** chromosomal start position, in base pairs

**end** chromosomal end position, in base pairs

**strand** chromosomal strand, + or -

**un** unknown

**ensemblID** ENSEMBL identifier

**type** type of transcript

**stat** status of transcript

**name** HUGO identifier

**entrezID** Entrez identifier

@source ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_mouse/release_M7/

---

auc_multifunc *Calculating AUC for functional groups from ranked lists*

---

## Description

The function calculates the AUC for a functional group analytically using an optimal ranked list of genes that indicates association between genes and groups.

## Usage

```
auc_multifunc(annotations, optimallist)
```

## Arguments

| | |
|---|---|
| annotations | binary matrix indicating which list elements are in which functional groups. |
| optimallist | Ranked list (multifunctionality analysis, see calculate_multifunc). |

## Value

aucs array of aucs for each group in annotations

## Examples

```
annotations <- c(rep(0,10))
annotations[c(1,3,5)] <- 1
optimallist <- 10:1
aurocs_mf <- auc_multifunc(annotations, optimallist)
```

## auprc *Area under the precision recall curve*

### Description

The function calculates the area under the precision-recall curve

### Usage

```
auprc(scores, labels)
```

### Arguments

| | |
|---|---|
| scores | numeric array |
| labels | binary array |

### Value

auprc Numeric value

### Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
auprc <- auprc(scores, labels)
```

## auroc_analytic *Area under the receiver operating characteristic curve*

### Description

The function calculates the area under the receiver operating characteristic (ROC) curve analytically

### Usage

```
auroc_analytic(ranks, labels)
```

### Arguments

| | |
|---|---|
| ranks | numeric array |
| labels | binary array |

### Value

auroc Numeric value

### Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
auroc <- auroc_analytic(scores, labels)
```

---

## biogrid *BIOGRID v3.4.126*

---

### Description

A data frame containing protein-protein interactions

### Format

A data frame with 211506 rows and 2 variables:

**entrezID_A** List of Entrez identifiers, interactor A

**entrezID_B** List of Entrez identifiers, interactor B

@source http://thebiogrid.org/

---

## build_binary_network *Builds a binary network*

---

### Description

The function creates a gene-by-gene matrix with binary entries indicating interaction (1) or no interaction (0) between the genes.

### Usage

```
build_binary_network(data, list)
```

### Arguments

| | |
|---|---|
| data | 2-column matrix, each row a pair indicating a relationship or interaction |
| list | string array of genes/labels/ids |

### Value

net matrix binary characterizing interactions

## Examples

```
data <- cbind(edgeA=c('gene1','gene2'),edgeB=c('gene3','gene3'))
list <- c('gene1','gene2','gene3')
network <- build_binary_network(data,list)
```

---

`build_coexp_expressionSet` *Builds a coexpression network from an expressionSet*

---

## Description

The function generates a dense coexpression network from expression data stored in the expressionSet data type. Correlation coefficicents are used as to weight the edges of the nodes (genes). Calls `build_coexp_network`.

## Usage

```
build_coexp_expressionSet(
  exprsSet,
  gene.list,
  method = "spearman",
  flag = "rank"
)
```

## Arguments

| | |
|---|---|
| `exprsSet` | data class ExpressionSet |
| `gene.list` | array of gene labels |
| `method` | correlation method to use, default Spearman's rho |
| `flag` | string to indicate if the network should be ranked |

## Value

net Matrix symmetric

## Examples

```
exprs <- matrix( rnorm(1000), ncol=10,byrow=TRUE)
gene.list <- paste('gene',1:100, sep='')
sample.list <- paste('sample',1:10, sep='')
rownames(exprs) <- gene.list
colnames(exprs) <- sample.list
network <- build_coexp_expressionSet(exprs, gene.list, method='pearson')
```

## build_coexp_GEOID *Builds a coexpression network given a GEO ID*

### Description

The function generates a dense coexpression network from expression data stored in GEO. The expression data is downloaded from GEO. Correlation coefficicents are used as to weight the edges of the nodes (genes). Calls `get_expression_matrix_from_GEO` and `build_coexp_network`.

### Usage

```
build_coexp_GEOID(gseid, gene.list, method = "spearman", flag = "rank")
```

### Arguments

| | |
|---|---|
| `gseid` | string GEO ID of expression experiment |
| `gene.list` | array of gene labels |
| `method` | correlation method to use, default Spearman's rho |
| `flag` | string to indicate if the network should be ranked |

### Value

net Matrix symmetric

## build_coexp_network *Builds a coexpression network from an expressionSet*

### Description

The function generates a dense coexpression network from expression data stored as a matrix, with the genes as row labels, and samples as column labels. Correlation coefficicents are used as to weight the edges of the nodes (genes). Calls `cor`.

### Usage

```
build_coexp_network(exprs, gene.list, method = "spearman", flag = "rank")
```

### Arguments

| | |
|---|---|
| `exprs` | matrix of expression data |
| `gene.list` | array of gene labels |
| `method` | correlation method to use, default Spearman's rho |
| `flag` | string to indicate if the network should be ranked |

## Value

net Matrix symmetric

## Examples

```
exprs <- matrix( rnorm(1000), ncol=10,byrow=TRUE)
gene.list <- paste('gene',1:100, sep='')
sample.list <- paste('sample',1:10, sep='')
rownames(exprs) <- gene.list
colnames(exprs) <- sample.list
network <- build_coexp_network(exprs, gene.list)
```

---

# build_semantic_similarity_network

*Builds a semantic similarity network*

---

## Description

The function builds a semantic similarity network given a data and labels

## Usage

```
build_semantic_similarity_network(genes.labels, genes)
```

## Arguments

| | |
|---|---|
| genes.labels | matrix with rows as genes and columns as a function/label |
| genes | array of gene IDs |

## Value

net Numeric value

## Examples

```
genes.labels <- matrix( sample(c(0,1), 100, replace=TRUE),ncol=10,nrow=10)
rownames(genes.labels) <- 1:10
genes <- 1:10
net <- build_semantic_similarity_network(genes.labels, genes)
```

build_weighted_network *Builds a weighted network*

## Description

The function creates a gene-by-gene matrix with binary entries indicating interaction (1) or no interaction (0) between the genes.

## Usage

```
build_weighted_network(data, list)
```

## Arguments

| | |
|---|---|
| data | 3-column matrix, each row a pair indicating a relationship or interaction, and the last column the weight |
| list | string array of genes/labels/ids |

## Value

net matrix characterizing interactions

## Examples

```
data <- cbind(edgeA=c('gene1','gene2'),edgeB=c('gene3','gene3'), weight=c(0.5, 0.9))
list <- c('gene1','gene2','gene3')
network <- build_weighted_network(data,list)
```

calculate_multifunc *Performing multifunctionality analysis*

## Description

The function performs multifunctionality analysis ([1]) for a set of annotated genes and creates a rank based optimallist. For annotations use an ontology that is large enough to serve as a prior (e.g. GO, Phenocarta).

## Usage

```
calculate_multifunc(genes.labels)
```

## Arguments

| | |
|---|---|
| genes.labels | Annotation matrix |

## Value

gene.mfs Returns matrix with evaluation of gene function prediction by given labels:

## Examples

```
genes.labels <- matrix( sample(c(0,1), 100, replace=TRUE),ncol=10,nrow=10)
rownames(genes.labels) = paste('gene', 1:10, sep='')
colnames(genes.labels) = paste('label', 1:10, sep='')
mf <- calculate_multifunc(genes.labels)
```

---

# conv_smoother *Plot smoothed curve*

---

## Description

The function plots a smoothed curve using the convolve function.

## Usage

```
conv_smoother(X, Y, window, raw = FALSE, output = FALSE, ...)
```

## Arguments

| | |
|---|---|
| X | numeric array |
| Y | numeric array |
| window | numeric value indicating size of window to use |
| raw | boolean |
| output | boolean |
| ... | other input into the plot function |

## Value

smoothed X,Y and std Y matrix

## Examples

```
x <- 1:1000
y <- rnorm(1000)
conv <- conv_smoother(x,y,10)
```

## example_annotations *Example of annotations*

### Description

This dataset includes

## example_binary_network *Example of binary network*

### Description

This dataset includes

### Format

Matrices and vectors

## example_coexpression *Example of binary network*

### Description

This dataset includes

### Format

Matrices and vectors

## example_neighbor_voting *Example of binary network*

### Description

This dataset includes

### Format

**entrezID** chromosomal start position, in base pairs
**name** HUGO gene identifier
**species** species
**disease** disease

## extend_network *Builds an extended network from a binary network*

### Description

The function extends a binary network by using the inverse of the path length between nodes as a weighted edge

### Usage

```
extend_network(net, max = 6)
```

### Arguments

| | |
|---|---|
| net | matrix binary and symmetric |
| max | numeric maximum number of jumps |

### Value

ext_net matrix dense and symmetric

### Examples

```
net <- matrix( sample(c(0,1),36, replace=TRUE), nrow=6,byrow=TRUE)
ext_net <- extend_network(net)
```

## filter_network *Filter on matrix*

### Description

The function filters out the rows or columns of a matrix such that the size of the group is exclusively between given min and max values

### Usage

```
filter_network(network, flag = 1, min = 0, max = 1, ids = NA)
```

### Arguments

| | |
|---|---|
| network | numeric matrix |
| flag | numeric 1 for row filtering, 2 for column filtering |
| min | numeric value |
| max | numeric value |
| ids | array to filter on |

## Value

network numeric matrix

## Examples

```
net <- matrix( rnorm(10000), nrow=100)
filt_net <- filter_network(net,1,10,100)
```

---

`filter_network_cols` *Filter on columns*

---

## Description

The function filters out the columns of a matrix such that the size of the group is exclusively between given min and max values

## Usage

```
filter_network_cols(network, min = 0, max = 1, ids = NA)
```

## Arguments

| | |
|---|---|
| `network` | numeric matrix |
| `min` | numeric value |
| `max` | numeric value |
| `ids` | array |

## Value

network numeric matrix

## Examples

```
genes.labels <- matrix( sample( c(0,1), 10000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:100, sep='')
genes.labels <- filter_network_cols(genes.labels,50,200)

genes.labels <- matrix( sample( c(0,1), 10000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:100, sep='')
genes.labels <- filter_network_cols(genes.labels,ids = paste('function', 1:20, sep=''))
```

`filter_network_rows` *Filter on rows*

## Description

The function filters out the rows of a matrix such that the size of the group is exclusively between given min and max values

## Usage

```
filter_network_rows(network, min = 0, max = 1, ids = NA)
```

## Arguments

| | |
|---|---|
| `network` | numeric matrix |
| `min` | numeric value |
| `max` | numeric value |
| `ids` | array to filter on |

## Value

network numeric matrix

## Examples

```
genes.labels <- matrix( sample( c(0,1), 10000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:100, sep='')
genes.labels <- filter_network_rows(genes.labels,50,200)

genes.labels <- matrix( sample( c(0,1), 10000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:100, sep='')
genes.labels <- filter_network_rows(genes.labels,ids = paste('gene', 1:20, sep=''))
```

## filter_orthologs *Filter on orthologs*

### Description

The function filters away the labels for the genes that are not in the orthologs list

### Usage

```
filter_orthologs(annotations, genelist, orthologs)
```

### Arguments

| | |
|---|---|
| annotations | binary matrix |
| genelist | array of gene ids |
| orthologs | array to filter on |

### Value

annotations_filtered binary matrix

### Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')
gene.list <- paste('gene', 1:100, sep='')
orthologs <- paste('gene', (1:50)*2, sep='')
genes.labels.filt <- filter_orthologs(genes.labels, gene.list, orthologs)
```

## fmeasure *Fmeasure of precision-recall*

### Description

The function calculates fmeasure for a given beta of a precision-recall curve

### Usage

```
fmeasure(recall, precis, beta = 1)
```

### Arguments

| | |
|---|---|
| recall | numeric array |
| precis | numeric array |
| beta | numeric value, default is 1 |

## Value

fmeasure Numeric value

## Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
prc <- get_prc(scores, labels)
fm <- fmeasure(prc[,1], prc[,2])
```

---

# genes *Genes from BIOGRID v3.4.126*

---

## Description

An array containing identifiers for genes in biogrid

## Format

Array

**genes** List of Entrez identifiers

@source http://thebiogrid.org/

---

# get_auc *Calculates the area under a curve*

---

## Description

The function calculates the area under the curve defined by x and y

## Usage

```
get_auc(x, y)
```

## Arguments

| | |
|---|---|
| x | numeric array |
| y | numeric array |

## Value

auc numeric value

## Examples

```
x <- 1:100
y <- 1:100
auc <- get_auc(x,y)
```

---

`get_biogrid` *Downloading and filtering BIOGRID*

---

## Description

The function downloads the specifed version of biogrid for a particular taxon

## Usage

```
get_biogrid(species = "9606", version = "3.5.181", interactions = "physical")
```

## Arguments

| | |
|---|---|
| `species` | numeric taxon of species |
| `version` | string of biogrid version |
| `interactions` | string stating either physical or genetic interactions |

## Value

biogrid data.frame with interactions

---

`get_counts` *Get counts*

---

## Description

The function formats the count distribution from the histogram function

## Usage

```
get_counts(hist)
```

## Arguments

| | |
|---|---|
| `hist` | histogram |

## Value

x,y

## Examples

```
x <- runif(1000)
counts <- get_counts( hist(x, plot=FALSE))
```

---

`get_density` *Get density*

---

## Description

The function formats the density distribution from the histogram function

## Usage

```
get_density(hist)
```

## Arguments

| | |
|---|---|
| `hist` | histogram |

## Value

array

## Examples

```
x <- runif(1000)
density <- get_density( hist(x, plot=FALSE))
```

---

`get_expression_data_gemma`
*Obtain expression matrix from the GEMMA database*

---

## Description

The function downloads and parses the expression matrix from the GEMMA database, specified by the GEO ID

## Usage

```
get_expression_data_gemma(gseid, filtered = "true")
```

## Arguments

| | |
|---|---|
| `gseid` | GEO ID of the expression experiment |
| `filtered` | flag to indicate whether or not the data is QC |

### Value

list of genes and the expression matrix

---

`get_expression_matrix_from_GEO` *Obtain expression matrix from GEO database*

---

### Description

The function downloads and parses the expression matrix from the GEO file specified by the GEO ID

### Usage

```
get_expression_matrix_from_GEO(gseid)
```

### Arguments

| | |
|---|---|
| `gseid` | GEO ID of the expression experiment |

### Value

list of genes and the expression matrix

---

`get_phenocarta` *Downloading and filtering Phenocarta*

---

### Description

The function downloads the latest version of phenocarta

### Usage

```
get_phenocarta(species = "human", type = "all")
```

### Arguments

| | |
|---|---|
| `species` | string |
| `type` | string |

### Value

data data.frame with phenocarta data

## get_prc *Build precision-recall curve*

### Description

The function calculates the recall and precision

### Usage

```
get_prc(ranks, labels)
```

### Arguments

| | |
|---|---|
| ranks | numeric array |
| labels | binary array |

### Value

recall,precision numeric arrays

### Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
ranks <- rank(scores)
prc <- get_prc(ranks, labels)
```

## get_roc *Build receiver operating characteristic curve*

### Description

The function calculates the FPR and TRPR for the receiver operating characteristic (ROC)

### Usage

```
get_roc(ranks, labels)
```

### Arguments

| | |
|---|---|
| ranks | numeric array |
| labels | binary array |

### Value

FPR,TPR numeric arrays

### Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
ranks <- rank(scores)
roc <- get_roc(ranks, labels)
```

---

## GO.human *GO - human*

---

### Description

A dataset of the gene GO associations

### Format

A data frame with 2511938 rows and 4 variables:

**name** gene symbol
**entrezID** entrez identifier
**GO** gene ontology term ID
**evidence** evidence code

@source http://geneontology.org/

---

## GO.mouse *GO - mouse*

---

### Description

A dataset of the gene GO associations

### Format

A data frame with 2086086 rows and 4 variables:

**name** gene symbol
**entrezID** entrez identifier
**GO** gene ontology term ID
**evidence** evidence code

@source http://geneontology.org/

## GO.voc *Gene ontology vocabulary*

### Description

A dataset of the gene ontology vocabulary

### Format

A data frame with 42266 rows and 3 variables:

**GOID** GO identifier
**term** GO description
**domain** GO domain

@source http://geneontology.org/

## make_annotations *Creating gene annotations*

### Description

The function annotates a list of genes according to a given ontology. It creates a binary matrix associating genes (rows) with labels (columns).

### Usage

```
make_annotations(data, listA, listB)
```

### Arguments

| | |
|---|---|
| data | 2-column matrix, each row a pair indicating a relationship or interaction |
| listA | string array of genes |
| listB | string array of labels/functions |

### Value

net matrix binary

### Examples

```
gene.list <- paste('gene', 1:100, sep='')
labels.list <- paste('labels', 1:10, sep='')
data <- matrix(0,nrow=100, ncol=2)
data[,1] <- sample(gene.list, 100, replace=TRUE)
data[,2] <- sample(labels.list, 100, replace=TRUE)
net <- make_annotations(data, gene.list, labels.list)
```

## make_genelist *Creating list of all genes in the data set.*

### Description

The function extracts the list of all genes in the data set

### Usage

```
make_genelist(gene_data_interacting)
```

### Arguments

| | |
|---|---|
| gene_data_interacting | 2-column matrix, each row a pair indicating a relationship or interaction |

### Value

list array of data labels

### Examples

```
gene.list <- paste('gene', 1:100, sep='')
data <- matrix(0,nrow=100, ncol=2)
data[,1] <- sample(gene.list, 50, replace=TRUE)
data[,2] <- sample(gene.list, 50, replace=TRUE)
genes <- make_genelist(data)
```

## make_gene_network *Creating gene-by-gene network*

### Description

The function creates a gene-by-gene matrix with binary entries indicating interaction (1) or no interaction (0) between the genes.

### Usage

```
make_gene_network(data, list)
```

### Arguments

| | |
|---|---|
| data | 2-column matrix, each row a pair indicating a relationship or interaction |
| list | string array of genes |

## Value

net matrix binary characterizing interactions

## Examples

```
gene.list <- paste('gene', 1:100, sep='')
data <- matrix(0,nrow=100, ncol=2)
data[,1] <- sample(gene.list, 100)
data[,2] <- sample(gene.list, 100)
net <- make_gene_network(data, gene.list)
```

---

# make_transparent *Make a color transparent (Taken from an answer on StackOverflow by Nick Sabbe)*

---

## Description

Make a color transparent (Taken from an answer on StackOverflow by Nick Sabbe)

## Usage

```
make_transparent(color, alpha = 100)
```

## Arguments

| | |
|---|---|
| color | color number, string or hexidecimal code |
| alpha | numeric transparency |

## Value

someColor rgb

---

# neighbor_voting *Evaluating Gene Function Prediction*

---

## Description

The function performs gene function prediction based on 'guilt by association' using cross validation ([1]). Performance and significance are evaluated by calculating the AUROC or AUPRC of each functional group.

### Usage

```
neighbor_voting(
  genes.labels,
  network,
  nFold = 3,
  output = "AUROC",
  FLAG_DRAW = FALSE
)
```

### Arguments

| | |
|---|---|
| genes.labels | numeric array |
| network | numeric array symmetric, gene-by-gene matrix |
| nFold | numeric value, default is 3 |
| output | string, default is AUROC |
| FLAG_DRAW | binary flag to draw roc plot |

### Value

scores numeric matrix with a row for each gene label and columns auc: the average area under the ROC or PR curve for the neighbor voting predictor across cross validation replicates avg_node_degree: the average node degree degree_null_auc: the area the ROC or PR curve for the node degree predictor

### Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')
net <- cor( matrix( rnorm(10000), ncol=100), method='spearman')
rownames(net) <- paste('gene', 1:100, sep='')
colnames(net) <- paste('gene', 1:100, sep='')

aurocs <- neighbor_voting(genes.labels, net, output = 'AUROC')

avgprcs <- neighbor_voting(genes.labels, net, output = 'PR')
```

---

## node_degree *Calculate node degree*

---

### Description

The function calculates the node degree of a network

### Usage

```
node_degree(net)
```

### Arguments

| | |
|---|---|
| net | numeric matrix |

### Value

node_degree numeric array

### Examples

```
net <- cor( matrix(rnorm(1000), ncol=10))
n <- 10
net <- matrix(rank(net, na.last = 'keep', ties.method = 'average'), nrow = n, ncol = n)
net <- net/max(net, na.rm=TRUE)
nd <- node_degree(net)
```

---

## ortho *Gene orthologs*

---

### Description

A list containing identifiers for the subsets of gene orthologs

### Format

List orthologs for 5 species

**dros** List of Entrez identifiers, Drosophila

**celeg** List of Entrez identifiers, C. elegans

**yeast** List of Entrez identifiers, Yeast

**mouse** List of Entrez identifiers, Mouse

**zf** List of Entrez identifiers, Zebrafish

@source http://useast.ensembl.org/index.html/

## pheno *Phenocarta*

### Description

A dataset of gene disease associations

### Format

A data frame with 142272 rows and 4 variables:

**entrezID** chromosomal start position, in base pairs

**name** HUGO gene identifier

**species** species

**disease** disease

@source http://www.chibi.ubc.ca/Gemma/phenotypes.html

## plot_densities *Plot densities*

### Description

The function plots multiple density curves and compares their modes

### Usage

```
plot_densities(
  hists,
  id,
  col = c("lightgrey"),
  xlab = "",
  ylab = "Density",
  mode = "hist"
)
```

### Arguments

| | |
|---|---|
| hists | list of histogram objects or density objects |
| id | string |
| col | color for shading |
| xlab | string x-axis label |
| ylab | string y-axis label |
| mode | flag indicating histogram or density |

## Value

null

## Examples

```
aurocsA <- density((runif(1000)+runif(1000)+runif(1000)+runif(1000))/4)
aurocsB <- density((runif(1000)+runif(1000)+runif(1000))/3)
aurocsC <- density(runif(1000))
hists <- list(aurocsA, aurocsB, aurocsC)
temp <- plot_densities(hists,'', mode='density')
```

---

# plot_density_compare *Plot density comparisons*

---

## Description

The function plots two density curves and compares their modes

## Usage

```
plot_density_compare(
  aucA,
  aucB,
  col = "lightgrey",
  xlab = "AUROC (neighbor voting)",
  ylab = "Density",
  mode = TRUE
)
```

## Arguments

| | |
|---|---|
| aucA | numeric array of aurocs |
| aucB | numeric array of aurocs |
| col | color of lines |
| xlab | string label |
| ylab | string label |
| mode | boolean to plot mode or mean |

## Value

null

## Examples

```
aurocsA <- (runif(1000)+runif(1000)+runif(1000)+runif(1000))/4
aurocsB <- runif(1000)
plot_density_compare(aurocsA, aurocsB)
```

---

`plot_distribution` *Plot distribution histogram*

---

## Description

The function plots a the distribution of AUROCs

## Usage

```
plot_distribution(
  auc,
  b = 20,
  col = "lightgrey",
  xlab = "",
  ylab = "Density",
  xlim = c(0.4, 1),
  ylim = c(0, 5),
  med = TRUE,
  avg = TRUE,
  density = TRUE,
  bars = FALSE
)
```

## Arguments

| | |
|---|---|
| `auc` | numeric aucs |
| `b` | array of breaks |
| `col` | color of line |
| `xlab` | string label |
| `ylab` | string label |
| `xlim` | range of values for xaxis |
| `ylim` | range of values for yaxis |
| `med` | boolean to plot median auc |
| `avg` | boolean to plot average auc |
| `density` | boolean |
| `bars` | boolena for barplot |

## Value

auc list and quartiles

## Examples

```
aurocs <- (runif(1000)+runif(1000)+runif(1000)+runif(1000))/4
d <- plot_distribution(aurocs)
```

---

`plot_network_heatmap` *Plot network heatmap*

---

## Description

The function draws a heatmap to visualize a network

## Usage

```
plot_network_heatmap(net, colrs)
```

## Arguments

| | |
|---|---|
| `net` | a numeric matrix of edge weights |
| `colrs` | a range of colors to plot the network |

## Value

null

## Examples

```
network <- cor(matrix( rnorm(10000), nrow=100))
plot_network_heatmap(network)
```

## plot_prc *Plot precision recall curve*

### Description

The function calculates the precision and recall and plots the curve

### Usage

```
plot_prc(scores, labels)
```

### Arguments

| | |
|---|---|
| scores | numeric array |
| labels | binary array |

### Value

prc numeric arrays

### Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
roc <- plot_prc(scores, labels)
```

## plot_roc *Plot receiver operating characteristic curve*

### Description

The function calculates the FPR and TRPR for the receiver operating characteristic (ROC) and plots the curve

### Usage

```
plot_roc(scores, labels)
```

### Arguments

| | |
|---|---|
| scores | numeric array |
| labels | binary array |

## Value

FPR,TPR numeric arrays

## Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
roc <- plot_roc(scores, labels)
```

---

`plot_roc_overlay` *Plot ROC overlay*

---

## Description

The function plots a density overlay of ROCs given the scores and labels

## Usage

```
plot_roc_overlay(scores.mat, labels.mat, nbins = 100)
```

## Arguments

| | |
|---|---|
| `scores.mat` | numeric array |
| `labels.mat` | numeric array |
| `nbins` | numeric value |

## Value

list of Z(matrix) and roc_sum (average ROC curve)

## Examples

```
genes.labels <- matrix( c(rep(1, 1000), rep(0,9000)), nrow=1000, byrow=TRUE)
rownames(genes.labels) = paste('gene', 1:1000, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')

scores <- matrix( rnorm(10000), nrow=1000)
scores <- apply(scores, 2, rank)
rownames(scores) = paste('gene', 1:1000, sep='')
colnames(scores) = paste('function', 1:10, sep='')

z <- plot_roc_overlay(scores, genes.labels)
```

## plot_value_compare *Plot value comparisons*

### Description

The function plots a scatter

### Usage

```
plot_value_compare(
  aucA,
  aucB,
  xlab = "AUROC",
  ylab = "AUROC",
  xlim = c(0, 1),
  ylim = c(0, 1)
)
```

### Arguments

| | |
|---|---|
| aucA | numeric array of aucs |
| aucB | numeric array of aucs |
| xlab | string label |
| ylab | string label |
| xlim | range of values for xaxis |
| ylim | range of values for yaxis |

### Value

null

## predictions *Performing Gene Function Prediction*

### Description

The function performs gene function prediction on the whole data set using the 'guilt by association'-principle ([1]).

### Usage

```
predictions(genes.labels, network)
```

### Arguments

| | |
|---|---|
| genes.labels | numeric array |
| network | numeric array symmetric, gene-by-gene matrix |

### Value

scores numeric matrix

### Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')
net <- cor( matrix( rnorm(10000), ncol=100), method='spearman')
rownames(net) <- paste('gene', 1:100, sep='')
colnames(net) <- paste('gene', 1:100, sep='')

preds <- predictions(genes.labels, net)
```

---

## repmat *Rep function for matrices*

---

### Description

The function generates a matrix by binding the columns and rows

### Usage

```
repmat(X, m, n)
```

### Arguments

| | |
|---|---|
| X | numeric matrix |
| m | numeric value, repeat rows m times |
| n | numeric value, repeat columns n times |

### Value

list of genes and the expression matrix

### Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
expand <- repmat( genes.labels, 1,2)
```

run_GBA *Performing 'Guilt by Association' Analysis*

## Description

The function runs and evaluates gene function prediction based on the 'guilt by association'-principle using neighbor voting (neighbor_voting) [1]. As a measure of performance and significance of results, AUCs of all evaluated functional groups are calculated.

## Usage

```
run_GBA(network, labels, min = 20, max = 1000, nfold = 3)
```

## Arguments

| | |
|---|---|
| network | numeric array symmetric, gene-by-gene matrix |
| labels | numeric array |
| min | numeric value to limit gene function size |
| max | numeric value to limit gene function size |
| nfold | numeric value, default is 3 |

## Value

list roc.sub, genes, auroc

## Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')
net <- cor( matrix( rnorm(10000), ncol=100), method='spearman')
rownames(net) <- paste('gene', 1:100, sep='')
colnames(net) <- paste('gene', 1:100, sep='')

gba <- run_GBA(net, genes.labels, min=10)
```

# Index